Abstract

A synchronous pipeline design is provided that includes a first predetermined number of fetch logic sections, or "stages", and a second predetermined number of execution stages. Instructions are retrieved from memory and undergo instruction pre-decode and decode operations during the fetch stages of the pipeline. Thereafter, decoded instruction signals are passed to the execution stages of the pipeline, where the signals are dispatched to other execution logic sections to control operand address generation, operand retrieval, any arithmetic processing, and the storing of any generated results. Instructions advance within the various pipeline fetch stages in a manner that may be independent from the way instructions advance within the execution stages. Thus, in certain instances, instruction execution may stall such that the execution stages of the pipeline are not receiving additional instructions to process. This may occur, for example, because an operand required for instruction execution is unavailable. It may also occur for certain instructions that require additional processing cycles. Even though instructions are not entering the execution stages, instructions may continue to enter the fetch stages of the pipeline until all fetch stages are processing a respective instruction. As a result, when normal instruction execution resumes within the execution stages of the pipeline, all fetch stages of the pipeline have been filled, and pre-decode and decode operations have been completed for those instructions awaiting the entry into the execution stages of the pipeline.